Article

Energy-Efficient Swarming Flight Formation Transitions Using the Improved Fair Hungarian Algorithm

SungTae Moon ^{1,2}, Donghun Lee ³, Dongoo Lee ² , Doyoon Kim ² and Hyochoong Bang ^{1,*}

¹ Aerospace Systems and Control Laboratory, Korea Advanced Institute of Science and Technology (KAIST), Deajon 34141, Korea; stmoon@kari.re.kr

² Korea Aerospace Research Institute (KARI), Deajon 34133, Korea; ldg810@kari.re.kr (D.L.); doyoonkim@kari.re.kr (D.K.)

³ School of Aerospace and Mechanical Engineering, Korea Aerospace University (KAU), Goyang-si 10540, Korea; donghlee@kau.ac.kr

* Correspondence: hcbang@kaist.ac.kr

Abstract: Recently, drone shows have impressed many people through a convergence of technology and art. However, these demonstrations have limited operating hours based on the battery life. Thus, it is important to minimize the unnecessary transition time between scenes without collision to increase operating time. This paper proposes a fast and energy-efficient scene transition algorithm that minimizes the transition times between scenes. This algorithm reduces the maximum drone movement distance to increase the operating time and exploits a multilayer method to avoid collisions between drones. In addition, a swarming flight system including robust communication and position estimation is presented as a concrete experimental system. The proposed algorithm was verified using the swarming flight system at a drone show performed with 100 drones.

Keywords: swarming flight system; Hungarian algorithm; RTK-GPS

Citation: Moon, S.; Lee, D.; Lee, D.; Kim, D.; Bang, H. Energy-Efficient Swarming Flight Formation Transitions Using the Improved Fair Hungarian Algorithm. *Sensors* **2021**, *21*, 1260. <https://doi.org/10.3390/s21041260>

Academic Editor: Baochang Zhang

Received: 2 December 2020

Accepted: 3 February 2021

Published: 10 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As interest in unmanned aerial vehicles (UAVs) increases, many applications utilizing UAVs have been developed in various fields, such as delivery services [1] and reconnaissance. In particular, drone shows impressed many people around the world through a convergence of technology and art [2]. These demonstrations have shown significant advances in swarming flight technology.

For the swarming flight, various studies have been conducted in real environments with simulation [3–5]. In the early stages of research on swarming flight in a real environment, most researches were conducted indoors [6,7]. In the case of indoor swarming flight, motion capture or images can be used for position estimation. With the development of various sensors, various teams attempted to demonstrate swarm flight outdoors [8–10]. Later, Intel presented the first successful demonstration with 100 drones, in collaboration with Ars Electronica [10], and later presented a drone show involving more than 1000 drones during the 2018 Pyeongchang Winter Olympics, which was published in the Guinness Book of World Records. However, the system including communication and scenario generation has not been revealed because the system was developed privately.

In addition, the drone shows have limited operating hours based on the battery life. Thus, it is important to minimize the unnecessary transition time between scenes without collision to increase operating time. This transition problem can be regarded as a classical linear sum assignment problem (LSAP). The LSAP is one of the most famous problems in combinatorial optimization. The objective in this problem is to optimally assign agents to tasks individually such that the minimum cost or maximum profit can be obtained.

To solve the LSAP, diverse algorithms have been developed, including the simplex algorithm (standard linear programming) [11–13], the Hungarian algorithm [14,15], a neural algorithm [16], and the auction algorithm [17]. Each algorithm has undergone further

development to improve its time complexity [18]. Among the various LSAP algorithms, the Hungarian algorithm is the well-known optimal solution.

Many researchers have performed studies on the LSAP based on the Hungarian algorithm for the task assignment of multiple robots such as drones. Smriti et al. proposed a distributed version of the Hungarian method to improve task assignment efficiency [19]. Xiangming et al. developed an intelligent target detection method of UAV swarms using the Hungarian algorithm-based path planning method and three dimensions (3D) real-time probability map which estimates the possibility of detecting new targets [20]. Amir et al. described the minimum cost drone-station matching algorithm considering energy and distance for a smart city [21]. Sarah et al. proposed the Decentralized Hungarian-Based Algorithm (DHBA) to improve computational time and the optimality of assignments [22].

As shown in Table 1, most algorithms based on the Hungarian method have attempted to find an optimal solution that minimizes or maximizes the weight or improves the time complexity. However, although these algorithms can find an optimal solution that minimizes the sum of the transition times for each drone, it does not guarantee the minimization of the transition time in the 7drone shows, as the transition is completed only after all the drones have completely moved to the next scene position. Even when most drones have arrived at their destinations, these drones must wait until the last drone arrives because all the drones must be deployed in their designated positions for the next scene. As a result, the Hungarian algorithm is not efficient in terms of drone battery usage, and consequently, the overall operating time is reduced.

Table 1. Comparison with previous works.

	Application	Algorithm ($\sigma(*)$)	Weight (w)	Method	Verification (# of Targets)
Xiangming et al. [20]	Path planning	Original Hungarian	Quality	$\max \sum_{i=1}^n w_{x_i y_{\sigma(i)}}$	Simulation (22)
Amir et al. [21]	Drone-station matching in Smart city	Original Hungarian	Energy	$\max \sum_{i=1}^n w_{x_i y_{\sigma(i)}}$	Simulation (500)
Smriti et al. [19]	Multi-robot orchestra	Distributed Hungarian	Distance	$\min \sum_{i=1}^n w_{x_i y_{\sigma(i)}}$	Demonstration (10)
Sarah et al. [22]	Fast and scalable task allocation	DHBA	Distance	$\min \sum_{i=1}^n w_{x_i y_{\sigma(i)}}$	Simulation (50)
Our works	Scene transition in Drone show	Fair Hungarian	Distance	$\min \{ \max_{i \in X, j \in Y} (w_{ij}) \}$	Demonstration (100)

In this paper, a Fair Hungarian algorithm is proposed to generate a scenario that includes transitions between scenes within a short time without collision. The goal of the proposed algorithm is to achieve fast and energy-efficient scene transitions while minimizing the transition times between scenes during the generation of a scenario.

The main contributions of this paper can be summarized as follows:

- (1) To increase the operating time for drone shows, the Fair Hungarian algorithm is proposed to achieve fair energy consumption. The proposed algorithm equalizes the energy demand of the drones by minimizing the maximum movement distance between drones in a swarming flight scenario.
- (2) The drone show technology stacked on the veil is discussed. In this paper, methods to realize efficient communication and reliable position estimation for a swarming flight system are discussed. The communication mechanism can operate regardless of the number of drones. The position estimation based on the real time kinematic global positioning system (RTK-GPS) can switch mode smoothly when the RTK-GPS is not used.

- (3) The algorithm and system are verified through implementation in drone shows involving 100 drones with numerical experiments.

The paper is organized as follows. Section 2 describes the overall architecture of the swarming flight system considering efficient communication and position estimation. Section 3 discusses how to generate a safe and efficient scenario. Section 4 discusses the results of swarming flight experiments, and finally, Section 5 presents the conclusions and future research plans.

2. System Architecture

2.1. Overall Architecture

A swarming flight system should be deployed as simply as possible to facilitate the operation of a large number of drones and frequent movements between show locations [23]. Therefore, the proposed swarming flight system is designed to require only a simple GCS to ensure convenience of operation. The swarming flight system consists of a GCS, an RTK-GPS base station, drones, and scenarios generated by the user, as shown in Figure 1. The proposed GCS can be implemented on a laptop computer because it does not require high performance equipment for command transmission. The base station is used to transmit the correction data to enable precise position estimation based on RTK-GPS sensors, which are more precise than global positioning system (GPS) sensors [24]. Using this system, position estimation at the centimeter level can be realized. Each drone is equipped with a PX4 flight control computer (FCC) [25,26] including various algorithms using position control, motion equations and electromagnetic theory [27–29] based on open-source software. Notably, the PX4 FCC [30] is modified to enable efficient communication as well as robust control and position estimation during the swarming flight. The scenario for a drone show is prepared by the user before the flight and stored in the drones. Finally, each drone communicates with the GCS using the micro air vehicle link (MAVLink) [31], which is a lightweight messaging protocol for communication via Wi-Fi.

Figure 1. Deployment of the proposed swarming flight system.

In an emergency situation, the GCS can detect and control each individual malfunctioning drone during the scenario-based flight. However, if the communication system of the GCS experiences problems and all communication is lost, the GCS cannot control the drones. In this case, all the drones can be controlled by a grouped remote controller and landed in the direct downward direction to protect spectators.

The main components of the drones and GCS are shown in Figure 2. The drone system consists of an RTK-GPS module, a Wi-Fi module, a light emitting diode (LED), and an FCC module. In the FCC, several components are added to the basic PX4 FCC module. Each component communicates with other components through the message-driven method of a micro object request broker (μ ORB) [26]. The μ ORB is designed following

a publish-subscribe model. The MAVLink component manages the communication between the GCS and drone by using the MAVLink protocol, which is optimized for the drone. The scenario component is operated in accordance with an already-generated scenario file, which is transmitted from the GCS. This component passes the position and LED commands to the position controller and LED component, respectively. Using a mode switching algorithm mentioned in Section 2.3, the position estimator component can robustly estimate the drone's position even if the fixed mode of the RTK-GPS module suddenly fails. The monitoring component mentioned in Section 2.2 checks the health of the drone and operates within the drone to reduce the information exchanged with the GCS.

Figure 2. Control architecture for swarming flight (solid box: contributed software component, dotted box: PX4 software component, dashed box: hardware module).

The ground system consists of the RTK-GPS base station, the GCS, and the scenario creator. The RTK-GPS base station transmits the RTK-GPS correction information to each drone. The GCS has several components. The MAVLink component is the same as the corresponding component on each drone. To check the status of each drone in real time, the swarm monitoring component monitors the status of each drone based on the information received from the drone. The scenario component transmits the scenario information provided by the scenario generator mentioned in Section 3 to each drone. The emergency control component controls the drones that experience problems during the scenario flight.

2.2. Efficient Communication

As the number of drones involved in swarming flight increases, communication limitations due to the bandwidth are encountered. In addition, although wireless routers and other access points can theoretically support up to approximately 250 connected devices, such a large number of drones cannot be simultaneously connected because radio interference among the Wi-Fi devices may deteriorate the network performance due to frequent rebroadcasting of the messages that fail to reach their destinations, eventually leading to connection drops [32].

Nevertheless, it is essential to continuously transmit the correction data to maintain the fixed RTK-GPS mode to enable precise position estimation during the swarming flight. If the RTK-GPS correction data stream stops for a long time due to burst packet loss caused by unexpected network congestion, the fixed mode will change to the floating mode. Therefore, it is important to maintain communication stability and to protect against burst packet loss to maintain the fixed RTK-GPS mode.

Fortunately, the fixed mode can be maintained for a limited time even if the transmission of the RTK-GPS correction data stops. To check the health status of the fixed mode,

the age of the correction information is used, as shown in Figure 3. When a drone receives the correction data, the age of the correction information is reset.

Figure 3. Age-of-correction status.

Various communication algorithms have been proposed in order to stably operate many drones at the same time [33,34]. However, such network systems cannot incorporate all drones for a large swarming flight system because of the limitations of Wi-Fi technology. In the Wi-Fi framework, the carrier sense multiple access with collision avoidance (CSMA/CA) mechanism is used to manage the collision of transmitted packets [35]. This can be used to manage undefined nodes that are accessed randomly. However, this mechanism cannot support reliable real-time communication because the throughput of Wi-Fi decreases dramatically when many drones are simultaneously connected [36]. To solve this problem, the amount of data to be exchanged in the proposed system is reduced as much as possible by removing unnecessary information. This approach allows for an unlimited number of drones to be operated using the proposed system.

To reduce the amount of data exchanged in the proposed system, the data transmitted from the GCS to the drones should be optimized. Normally, GCS sends a command to the drone, and the drone sends the drone's status to the GCS. To compare the performance, we assume the same environment. The GCS transmits the correction for RTK-GPS and scenario at 1 and 10 Hz. The drones transmit the drone's status at 1 Hz. In the traditional swarming flight system which is depicted in Figure 4, the communication traffic (d_{s1}) increases linearly with the number of drones, as follows:

$$d_{s1} = n \times (d_{base} \times 1 + d_{scen} \times 10 + d_{status} \times 1) \quad (1)$$

where d_{base} and d_{scen} denote the amounts of correction data and scenario data, respectively, transmitted from the GCS to a drone. d_{status} denotes the amount of status data, such as battery data, transmitted from a drone to the GCS and n is the number of drones. This approach is useful for unknown missions such as collision avoidance in response to some event, because the drones can be controlled in real time. However, it is not suitable for drone shows that require a certain amount of traffic regardless of the number of drones.

In the proposed system, to reduce the communication traffic, the scenario information, which has already been prepared, is transmitted to and stored on each drone before the flight, as shown in Figure 5. In addition, the correction data from the RTK-GPS base station are transmitted through broadcasting. Consequently, the amount of data transmitted from the GCS to the drones is constant, as follows:

$$d_{s2} = d_{base} \times 1 + n \times d_{status} \quad (2)$$

Figure 4. Traditional communication approach for the swarming flight.

Figure 5. Proposed communication approach for the swarming flight.

However, if each drone needs to transmit several pieces of information, such as the sensor data for health status monitoring, the amount of data transmitted from the drones to the GCS will increase in proportion to the number of drones. To minimize the amount of data transmitted in this direction, in the proposed swarming flight system, the health status of each drone is not checked using the ground station system. Instead, the health status of each drone is checked at the drone by using a health decision module, as shown in Figure 6. The health decision module checks whether the drone is healthy in terms of each considered status indicator and expresses each status result as a bit. In addition, to further minimize the information sent from the drones to the ground station, this information is allowed to be transmitted only when changes occur.

To verify the proposed communication system, an experiment was conducted over a continuous period of 360 seconds based on a scenario with 100 drones. Figure 7 shows the maximum and average packet losses from the GCS to the drones for the RTK-GPS correction data as functions of the distance. In most cases, only one packet was dropped. In terms of the maximum packet loss, up to 4 packets were dropped. Therefore, the proposed system could maintain the fixed RTK-GPS mode.

Figure 6. Comparison of monitoring methods.

Figure 7. Results of the maximum and average burst packet loss.

2.3. Position Estimation

The proposed system utilizes RTK-GPS sensors in combination with inertial measurement units (IMU) and barometric sensors to realize accurate position estimation. In the RTK-GPS technique, measurements of the phase of the signal's carrier wave are used in addition to the information content of the signal, and a base station or interpolated virtual station provides real-time corrections to achieve up to centimeter-level accuracy [24]. However, if the satellite signal is weak or the communication conditions are inferior, the precise signal may be missed, or the incorrect value may be reported. In a swarming flight system, this situation is undesirable because the drones are located close to one another and are likely to collide. To overcome this problem, a mode switching algorithm is proposed for the swarming flight system to change the sensor fusion mode depending on the RTK-GPS conditions. When the RTK-GPS conditions are inferior, the GPS mode, in which the GPS information is mainly used along with the IMU and barometer data, is applied. Once the RTK-GPS conditions improve, the mode is switched back from the GPS to RTK-GPS. The objective is for the RTK-GPS mode to be used as much as possible.

Notably, when the mode changes, especially from the GPS to RTK-GPS mode, several position errors may accumulate in the estimation due to the GPS sensor error. This phenomenon may lead to sudden position changes, which may cause the spectators of the drone show to feel insecure. To solve this problem, a modified position estimation algorithm is proposed to enable smooth prediction of the future positions. The proposed position estimation algorithm is designed to predict the position while smoothly vary-

ing the information ratio (ρ) when switching from the GPS information to the RTK-GPS information, as shown in Figure 8.

Figure 8. Smooth mode switching.

To implement this approach, the modified position estimation algorithm is applied to enhance the position estimator module of the PX4 FCC [37]. Using the position and velocity from sensors are measured, the position and velocity information are predicted through the accelerometer, as follows:

$$\bar{p}_k = \hat{v}_{k-1}\Delta t + \frac{1}{2}a_{k-1}\Delta t^2 + \hat{p}_{k-1} \quad (3)$$

$$\bar{v}_k = a_{k-1}\Delta t + \hat{v}_{k-1} \quad (4)$$

where $\bar{p}_k$ and $\bar{v}_k$ denote the predicted position and velocity at k^{th} time. $\hat{p}_{k-1}$ and $\hat{v}_{k-1}$ indicate the corrected position and velocity. a_{k-1} is the value of acceleration. The predicted value is compared with the information received from RTK-GPS and GPS sensor and each error is measured, as follows:

$$e_{p,k}^{(i)} = z_{p,k}^{(i)} - \bar{p}_k \quad (5)$$

$$e_{v,k}^{(i)} = z_{v,k}^{(i)} - \bar{v}_k \quad (6)$$

where i^{th} is sensor number (1: GPS, 2: RTK-GPS). $e_{p,k}^{(i)}$ and $e_{v,k}^{(i)}$ indicate position and velocity error of i^{th} sensor between measured and predicted value at k^{th} time. $z_{p,k}^{(i)}$ and $z_{v,k}^{(i)}$ are measured position and velocity respectively.

After that, the correction of the position and velocity vectors was made using the position and velocity errors in Equations (5) and (6), as follows

$$\hat{p}_k = \bar{p}_k + \sum_i e_{p,k}^{(i)} w_p^{(i)} \rho^{(i)} \Delta t \quad (7)$$

$$\hat{v}_k = \bar{v}_k + \sum_i e_{v,k}^{(i)} w_v^{(i)} \rho^{(i)} \Delta t \quad (8)$$

where $w_p^{(i)}$ and $w_v^{(i)}$ denote weight of position and velocity of i^{th} sensor.

Then, the accelerometer bias is updated with the error value, as follows:

$$a_{k+1} = a_k - \sum_i R_E^B (e_{p,k}^{(i)} w_p^{(i)2} + e_{v,k}^{(i)} w_v^{(i)}) \rho^{(i)} \Delta t \quad (9)$$

where R_E^B is a transformation matrix from global to body coordinate system.

As a result of the mode switching algorithm, as shown in Figure 9, even if an error occurred due to a problem in the RTK-GPS position information, the affected drone will return smoothly to its trajectory.

Figure 9. Mode switching results.

3. Scenario Generation

A scenario for a swarm of drones includes the trajectory of each drone at any given time. A scenario is typically divided into several scenes to be displayed in the sky. When creating scenarios, the transitions between the scenes must be considered. To ensure a safe transition, the trajectory of each drone should be designed to avoid collisions. The sum of the movement distances of the trajectories should be minimized to ensure rapid transition. Moreover, the total movement distance of each drone should be similar to ensure energy efficiency. The proposed swarming flight system can ensure safe and energy-efficient scene transitions while minimizing the transition times between the scenes during the creation of a scenario. In addition, the system can land the drones safely even when several drones encounter difficulties.

3.1. Problem Statement

To enable an efficient display, the scenes should transition as rapidly as possible. The problem of designing the drone movement for a scene transition is similar to an assignment problem in a bipartite graph whose vertices can be divided into two disjoint and independent sets X and Y such that every edge connects a vertex in X to an edge in Y , as shown in Figure 10 [38].

Figure 10. Scene transition (left: based on spatial coordinates, right: based on a bipartite graph).

To represent the problem, let $G = (X \cup Y, E)$ be a bipartite graph, where X and Y represent the drones in the current scene and next scene respectively. x_i denotes the i^{th} vertex in X and y_j denotes the j^{th} vertex in Y , where $1 \leq i \leq |X|$ and $1 \leq j \leq |Y|$ [38,39]. $|X|$ and $|Y|$ denote the numbers of elements in X and Y , respectively. In this problem, $|X|$ and $|Y|$ are equal because the same number of drones is used throughout a scenario. $w(e_{x_i y_j})$ denotes nonnegative weight of $e_{x_i y_j}$ which is the edge connected between vertex x_i and y_j . In this problem, the weight indicates the Euclidian distance between drone x_i of the current scene X and drone y_j of the next scene Y , as follows:

$$w(e_{x_i y_j}) = \sqrt{(p_x^{x_i} - p_x^{y_j})^2 + (p_y^{x_i} - p_y^{y_j})^2 + (p_z^{x_i} - p_z^{y_j})^2} \quad (10)$$

where $(p_x^{x_i}, p_y^{x_i}, p_z^{x_i})$ and $(p_x^{y_j}, p_y^{y_j}, p_z^{y_j})$ the 3D position of drone x_i and y_j , respectively.

All the edge weights in the graph are represented using a $|X| \times |Y|$ matrix C . This matrix denotes edge set in the complete bipartite graph whose vertices can be partitioned into two subsets, drones of current scene and next scene, as follows:

$$C = \begin{bmatrix} w(e_{x_1 y_1}) & \dots & w(e_{x_1 y_n}) \\ \vdots & \ddots & \vdots \\ w(e_{x_n y_1}) & \dots & w(e_{x_n y_n}) \end{bmatrix} \quad (11)$$

Therefore, the given problem is to match x_i to y_j such that each vertex in X is connected to exactly one vertex in Y while minimizing the sum of the costs. In other words, the problem is to find a function $\sigma(i)$, representing the index of a vertex, such that the sum of the cost functions is minimized, as follows:

$$\min \sum_{i=1}^n w_{x_i y_{\sigma(i)}} \quad (12)$$

3.2. Hungarian Algorithm

To find $\sigma(i)$, the Hungarian algorithm, also known as the Kuhn-Munkres algorithm, is generally adopted [14,39–41]. This algorithm is an optimization method to solve the assignment problems to maximize or minimize a cost. In this paper, the Hungarian algorithm is used for cost minimization. We proceed to provide a brief description of the Hungarian algorithm to assist explaining the proposed algorithm. Algorithm 1 presents the Hungarian algorithm in the form of pseudocode.

Algorithm 1. Pseudocode for the Hungarian algorithm Function Hungarian-Algorithm (G).

```

% initial vertex feasible labeling  $l_1$ 
 $i = 1, j = 1$ 
 $\forall x \in X, l_1(x) = \min_{y \in Y} \{w(e_{xy})\}$ 
 $\forall y \in Y, l_1(y) = 0$ 
 $M_1 \leftarrow$  initial matching
while  $|M_i| < |X|$ 
    % find the maximum matching  $M$ 
     $M_i^j = M_i$ 
    do
         $F_i^{j+1} \leftarrow$  alternating forest of  $M_i^j$  rooted at the unmatched vertices  $x \in X$ 
         $P_i^{j+1} \leftarrow$  a path of  $F_i^{j+1}$  which contain another unmatched vertex  $y \in Y$ 
         $M_i^{j+1} = M_i^j \oplus E(P_i^{j+1})$ 
         $Q_i^{j+1} = [X - V(F_i^{j+1})] \cup [Y \cap V(F_i^{j+1})]$ 
         $j = j + 1$ 
    while  $|M_i^j| \neq |Q_i^j|$ 
     $M_i = M_i^j$ 
    % update the feasible vertex labeling
     $F_i \leftarrow$  alternating forest of  $M_i$  rooted at the unmatched vertices  $x \in X$ 
     $S = V(F_i) \cap X$ 
     $T = V(F_i) \cap Y$ 
     $\alpha = \min_{x \in S, y \notin T} \{w(e_{xy}) - l_i(x) - l_i(y)\}$ 
     $l_{i+1}(v) = \begin{cases} l_i(v) + \alpha & \text{if } v \in S \\ l_i(v) - \alpha & \text{if } v \in T \\ l_i(v) & \text{otherwise} \end{cases}$ 
     $i = i + 1$ 
done

```

3.3. Proposed Algorithm

Although the Hungarian algorithm can find the minimum sum of the costs, an efficient transition between the scenes cannot be ensured because a scene transition is complete only once all drones move completely to the next scene position. Even when most drones have arrived at their destinations, the drones must wait until the last drone arrives because all the drones need to be deployed in their designated positions for the next scene. Consequently, the Hungarian algorithm is not efficient in terms of drone battery usage, and consequently, the overall operating time is reduced. Moreover, it is essential to minimize the maximum movement distance among the drones in a swarming flight scenario. In this paper, a new efficient assignment algorithm is proposed to minimize the maximum weight (distance) before the Hungarian algorithm is applied, as follows:

$$\min\left\{\max_{x \in X, y \in Y} w(e_{xy})\right\} \quad (13)$$

To reduce the maximum weight, the cost matrix is updated by continuously removing the highest weight. However, an ideal matching may not be found in the bipartite graph if excessively many edges are removed. Therefore, the Hall theorem [42] is used to check that the perfect matching condition in a bipartite graph with $|X| = |Y|$ is satisfied. The bipartite graph G contains an ideal matching if and only if the Hall condition is satisfied, as follows:

$$|S| \geq |N(S)| \text{ for every } S \subseteq X \quad (14)$$

where $N(S)$ denotes the neighbors of S and can be expressed, as follows:

$$N(S) = \{y \in Y | e_{xy} \in E \text{ for some } x \in S\} \quad (15)$$

In addition, the binary search algorithm, also known as half-interval search, is used to reduce the complexity. The binary search algorithm finds the position of a target value within a sorted set of weights W . This algorithm checks whether E_o contains an ideal matching, according to the Hall theorem, up to the middle point m of the array. E_o is the optimized set of edges in which all the edges having weights greater than the m^{th} weight in the sorted weight set W have been removed, as follows:

$$E_o = E - \{e_{xy} | w(e_{xy}) \geq W_m, x \in X, y \in Y\} \quad (16)$$

where W_m denotes the m^{th} component of W .

If E_o contains an ideal matching, r is moved to $m - 1$ to eliminate the edges with weights greater than the middle point of the weight set. If an ideal matching is not identified, l is moved to m to seek an ideal matching. This loop is repeated until l is the same as r . In this manner, the minimum sum of the distances is optimized while reducing the standard deviation of the distances. The pseudocode for this algorithm is presented in Algorithm 2.

Once assignment matching has been performed, the drones should be separated into several layers to avoid collisions while the drones are moving. Each layer is located in a different plane, which is tilted with respect to the ground, to establish separate movement paths for the transition between the scenes. To detect collisions between the drones, the intersections between the movement paths should be checked. In this paper, a movement path is defined by its start and end points and is treated simply as a line segment, i.e., we assume that each path is straight [43]. A line segment is a part of a line that is bounded by two distinct points, specifically, the start point (p_0) and end point (p_1). The points along the line segment can be represented in terms of $u \in [0, 1]$, as follows:

$$p = p_0 + u(p_1 - p_0) \quad (17)$$

To identify an intersection between two line segments l and m , the intersection point, where $p^l = p^m$, is calculated, as follows:

$$p_0^l + u^l(p_1^l - p_0^l) = p_0^m + u^m(p_1^m - p_0^m) \quad (18)$$

Each point p can be expressed in terms of its coordinates (x, y) in a plane. Solving for the intersection point yields the following two equations in terms of two unknowns, u^l and u^m , as follows:

$$x_0^l + u^l(x_1^l - x_0^l) = x_0^m + u^m(x_1^m - x_0^m) \quad (19)$$

$$y_0^l + u^l(y_1^l - y_0^l) = y_0^m + u^m(y_1^m - y_0^m) \quad (20)$$

Algorithm 2. Pseudocode for the proposed Fair Hungarian algorithm

```

G = (X ∪ Y, E)
W = sort({w(exy) | x ∈ X, y ∈ Y})
l = 1
r = |W|
while l ≠ r
    m = ceil((l+r)/2)
    Eo = E - {exy | w(exy) > W(m), x ∈ X, y ∈ Y}
    if |S| ≥ |N(S)| for S ⊆ X
        r = m - 1
    else
        l = m
done
Eo = E - {exy | w(exy) > W(l), x ∈ X, y ∈ Y}
Ĝ = (X ∪ Y, Eo)
M = Hungarian-Algorithm (Ĝ)
L = Layer (M)

```

u^l and u^m can be calculated, as follows:

$$u^l = \frac{(x_1^m - x_0^m)(y_0^l - y_0^m) - (y_1^m - y_0^m)(x_0^l - x_0^m)}{(y_1^m - y_0^m)(x_1^l - x_0^l) - (x_1^m - x_0^m)(y_1^l - y_0^l)} \quad (21)$$

$$u^m = \frac{(x_1^l - x_0^l)(y_0^l - y_0^m) - (y_1^l - y_0^l)(x_0^l - x_0^m)}{(y_1^m - y_0^m)(x_1^l - x_0^l) - (x_1^m - x_0^m)(y_1^l - y_0^l)} \quad (22)$$

If an intersection point between l and m exists, then u^l and u^m should satisfy the conditions $u^l \in [0, 1]$ and $u^m \in [0, 1]$ respectively.

However, even if no intersection exists between l and m , a collision may still occur in practice because of movement path errors or the influence of the finite drone size. Thus, a collision margin (ϵ) between l and m should be implemented.

Moreover, even if no intersection exists, a collision at the margin may occur at the start or end point. To detect a collision within the margin space, the minimum distance between the point and line is calculated at both (l, p_0^m) and (l, p_1^m) .

In the case of (l, p_0^m) , p_0^m is closest to the line at the tangent to l that passes through p_0^m . In other words, the dot product of the tangent and line is zero, as follows:

$$(p_0^m - p^l) \cdot (p_1^l - p_0^l) = 0 \quad (23)$$

$$(p_0^m - p_0^l - u^l(p_1^l - p_0^l)) \cdot (p_1^l - p_0^l) = 0 \quad (24)$$

$$(p_0^m - p_1^l) \cdot (p_1^l - p_0^l) - u^l(p_1^l - p_0^l) \cdot (p_1^l - p_0^l) = 0 \quad (25)$$

$$u^l = \frac{(p_0^m - p_1^l) \cdot (p_1^l - p_0^l)}{(p_1^l - p_0^l) \cdot (p_1^l - p_0^l)} \quad (26)$$

If $u^l < \epsilon$ when $u^l \in [0, 1]$, a collision may occur.

To separate the layers, all the drone movement paths are iteratively checked for collisions. If no collision occurs among a set of paths, the paths can be included in the same layer.

Moreover, in the proposed scene representation method, the scene is tilted by approximately 30° for safe realization, as shown in Figure 11. This configuration can minimize the collisions with other drones when a drone falls or must land in an emergency situation. In addition, compared to the right-angle (vertical) view, the tilted view allows the desired pattern to be displayed more accurately with respect to the viewing angle of the spectators.

Figure 11. Proposed scene representation method.

3.4. Numerical Example

Numerical experiments are conducted to illustrate the quality of the theoretical results and to compare the proposed Fair Hungarian algorithm with the original Hungarian algorithm. The numerical example focuses on the assignment problem. The numerical experiments were conducted by considering a scene transition involving 4 drones in each scene, as shown in Figure 12.

	y_1	y_2	y_3	y_4
x_1	1	5	9	7
x_2	5	9	8	9
x_3	8	9	5	9
x_4	9	8	1	6

Figure 12. Example used for the numerical experiments.

For the comparison, we first consider the original Hungarian algorithm presented in Algorithm 1. First, the initial vertex feasible labeling $l_1(x)$ is calculated with the minimum weights, as shown in Figure 13a.

Figure 13. Example for the numerical experiment ($i = 1, j = 1$).

Based on the initial vertex feasible labeling $l_1(x)$, the initial matching M_1 is greedily selected, as shown in Figure 13b. In the first iteration ($i = 1$), no longest path P that contains another unmatched vertex y in the alternating forest exists that can be chosen to construct the maximum matching because no alternating path exists. Consequently, the size of the maximum matching is the same as that of the vertex cover Q , as follows:

$$M_1^1 = \{x_1y_1, x_4y_4\}, |M_1^1| = 2 \quad (27)$$

$$Q = \{y_1, y_3\}, |Q| = 2 \quad (28)$$

Once the maximum matching has been found, the feasible vertex labeling is updated using the alternating forest of M_1 , as shown in Figure 13c. To find the vertex that will be updated for the next iteration, the sets S and T are identified based on the alternating forest. Subsequently, the feasible vertex labeling is updated using α , as shown in Figure 14a.

$$S = \{x_1, x_2, x_3, x_4\} \quad (29)$$

$$V = \{y_1, y_3\} \quad (30)$$

$$\alpha = \min_{x \in S, y \notin T} \{w(e_{xy}) - l_i(x) - l_i(y)\} = 4 \quad (31)$$

Figure 14. Example for the numerical experiment ($i = 2, j = 1$).

In the second iteration ($i = 2$), the matching M_2 is as shown in Figure 14b. The alternating forest based on the bipartite graph is shown in Figure 14c. The longest path in the forest is determined, as follows:

$$P = x_2y_1x_1y_2 \quad (32)$$

$$E(P) = \{x_2y_1, x_1y_1, x_1y_2\} \quad (33)$$

$$M_2^1 = \{x_1y_2, x_2y_1, x_4y_3\}, |M_2^1| = 3 \quad (34)$$

$$Q = \{y_1, y_2, y_3, y_4\}, |Q| = 4 \quad (35)$$

In this case, the matching is not an ideal matching because the number of edges in the matching, $|M_2^1|$, is not the same as in the vertex cover, $|Q|$. Therefore, the loop to determine the maximum matching must be iterated as shown in Figure 15.

$$P = x_3y_2x_1y_1x_2y_4 \quad (36)$$

$$E(P) = \{x_3y_2, x_1y_2, x_1y_1, x_2y_4, x_2y_1\} \quad (37)$$

$$M_2^2 = \{x_1y_1, x_2y_4, x_3y_2, x_4y_3\}, |M_2^2| = 4, \quad (38)$$

$$Q = \{y_1, y_2, y_3, y_4\}, |Q| = 4 \quad (39)$$

Figure 15. Example for the numerical experiment ($i = 2, j = 2$).

Finally, the matching M_2^2 is an ideal matching because $M_2^2 = |Q| = 4$. Consequently, the total sum of the weights is 20. However, the variance is large, as shown in Figure 16a. In other words, certain drones consume considerably more energy than other drones, shortening the overall operating time.

Figure 16. Comparison between the results of the Hungarian and Fair Hungarian algorithms.

In contrast, the proposed algorithm can reduce the variance, allowing the proposed system to achieve fair energy consumption and increasing the operating time of the drones as shown in Figure 16b. To reduce the variance, the cost matrix is first optimized in the proposed Fair Hungarian algorithm as presented in Algorithm 2. Before the cost matrix is updated, the weights are sorted, as follows:

$$W = \{1, 1, 5, 5, 5, 6, 7, 8, 8, 8, 9, 9, 9, 9, 9\} \quad (40)$$

Subsequently, the weights are removed by using the binary search algorithm until the Hall condition for the ideal matching is violated. Finally, the remaining weights are updated, as follows:

$$W = \{1, 1, 5, 5, 5, 6\} \quad (41)$$

Once the weights are updated, the ideal matching M is identified using the Hungarian algorithm based on the cost matrix obtained after the removal of weights. The proposed algorithm reduces the variance to 0.5, even though the total sum of the weights is 21, which is similar to the optimized minimum cost, as shown in Figure 16b.

4. Results of a Swarming Flight Experiment

4.1. Implementation

4.1.1. Drones

Since multiple drones are operated simultaneously in a swarming flight system, the drones should be constructed to be as simple as possible. Therefore, the proposed drone system is designed using simple modules, which are integrated in an interface board. The internal modules of each drone consist of an FCC module, an LED module, an RTK-GPS module, and a communication module.

It is assumed that in the swarming flight operation, the scenario flight requires approximately 10 min, and 5 min are required for the automatic initial deployment and initial operation tests. Thus, the proposed drone as shown in Figure 17 is designed to fly for 15 min. In the comparison with the Intel drone, the speed and flight time of the proposed drone are approximately two times those of the Intel drone, and thus the proposed drone is more tolerant to wind and can support a longer demonstration time, as indicated in Table 2.

Figure 17. Swarming flight drones (left: deployed drones, right: swarming flight drone).

Table 2. Performance comparison of the Intel Shooting Star and the proposed drone.

	Shooting Star [44]	Proposed Drone (Figure 17)
Type	quadcopter	Quadcopter
Size	$38.4 \times 38.4 \times 9.3$ cm	$50 \times 41 \times 16$ cm
Weight	330 g	800 g
Flight time	5–8 min	10–15 min
Maximum speed	7 m/s	19.4 m/s
Rotor diameter	6 inches	8 inches

4.1.2. Ground Control System (GCS)

For swarming flight operation, the GCS is implemented based on the Qt toolkit, which is an operating system independent framework, as shown in Figure 18. This system is designed to manage various heterogeneous drones to allow the use of different types of drones.

The architecture of the GCS is shown in Figure 18a. The system is divided into three layers: the user layer, manager layer, and agent layer. Each layer is independent of the other

layers and communicates with the other layers by using a message queue. Consequently, a legacy layer can be easily replaced with a new layer. The user layer manages a graphical user interface (GUI). This system can illustrate the drones in 3-dimensional space in real time, as shown in Figure 18b. The control layer manages all the drones and implements the specified scenarios. In an emergency situation, the GCS can control each drone as required in the given layer. The agent layer manages the communication and sends commands to the drones. This agent module should be developed differently for different types of drones.

Figure 18. GCS architecture (a) and GUI (b).

4.2. Experiment

To verify the proposed scene transition algorithm, a scenario consisting of 6 scenes was created. The scenes were named S1('init'), S2('3.1'), S3('100'), S4('flag'), S5('Korea'), and S6('KARI'). This scenario dataset is generated as an extensible markup language (XML) file and available at https://github.com/stmoon/swarm_flight_scenario (accessed on 23 January 2021). This scenario dataset includes the three-dimensional position (x, y, z) and yaw(ϕ) of each drone. Before operating the drone show, the drones are deployed as 10×10 like as shown in Figure 17a. Each drone which has a unique ID is initially placed 3m apart. With drone #1 as the origin, each drone has a relative position. For the experiment, a maximum of movement speed is limited by 4 m/s. The distance between drones in scenario was restricted to 1 m. The GCS is performed on a laptop computer with Intel(R) Core™ i7 CPU [45]. Each position of the drone is calculated using RTK-GPS.

To demonstrate the swarming flight formation, the drone show is operated at night as shown in Figure 19. It is performed during about 400 second for 6 scenes. The experimental result video is available at <https://youtu.be/TgCKhgljWW8> (accessed on 23 January 2021).

Figure 19. Drone show results.

4.3. Experimental Analysis

To compare the proposed algorithm with the Hungarian method, Figure 20 shows the direction of movement of each drone during each scene transition pertaining to the original Hungarian algorithm (left panels) and the proposed algorithm (center panels) along with the graphs (right panels) comparing the maximum distance for each drone.

Figure 20. Scene transition.

Figure 21 compares the sum of the distances, maximum movement distance among the drones, and standard deviation among the drones corresponding to the two Hungarian methods through boxplot. The proposed algorithm reduces the maximum movement distance among the drones, while the sum of the distances remains similar. In addition,

the overall battery consumption of the drones is more efficient because the standard deviation is smaller than that achieved using the original Hungarian method. Consequently, even though the Hungarian algorithm can find the assignment with the minimum total cost, the energy consumption is not fairly distributed among all the drones. In contrast, the proposed algorithm can assign fair movement paths while maintaining a similar cost.

Figure 21. Results of the improved Hungarian method.

In addition, each scene is separated into several layers to avoid collisions while the drones are moving. For example, in the case of the scene transition from S1 to S2, 4 layers are used, as shown in Figure 22.

Figure 22. Layers for the second scene. (S1→S2).

5. Conclusions

For the efficient swarming flight formation transition of drone show, the Fair Hungarian algorithm is proposed to achieve fair energy consumption and thus increase the operating time of drones in swarm flight. The proposed algorithm equalizes the energy demand of drones by minimizing the maximum movement distance between drones in a swarming flight scenario. In addition, a robust and efficient swarming flight system is discussed including the method of efficient communication and robust position estimation. This algorithm and system have been verified through swarming flight experiments with 100 drones.

The swarming flight system proposed to date has been applied for artistic purposes. However, the system can be applied to scout and monitor the large areas within a short time. In this case, the Fair Hungarian algorithm will be used to extend the mission time by switching drones without collision.

Author Contributions: Conceptualization, S.M. and H.B.; software, S.M. and D.L. (Dongoo Lee); validation, S.M., D.K. and D.L. (Dongoo Lee); formal analysis, S.M. and D.L. (Donghun Lee); investigation, S.M.; writing—original draft preparation, S.M. and D.L. (Donghun Lee); writing—review and editing, S.M., H.B. and D.L. (Donghun Lee); visualization, S.M.; supervision, H.B.; project administration, S.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF) and Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science and ICT, the Republic of Korea (2020M3C1C1A01083).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The scenario data presented in this study are available at https://github.com/stmoon/swarm_flight_scenario (accessed on 23 January 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kuru, K.; Ansell, D.; Khan, W.; Yetgin, H. Analysis and Optimization of Unmanned Aerial Vehicle Swarms in Logistics: An Intelligent Delivery Platform. *IEEE Access* **2019**, *7*, 15804–15831. [\[CrossRef\]](#)
2. Kshetri, N.; Rojas-Torres, D. The 2018 Winter Olympics: A Showcase of Technological Advancement. *IT Prof.* **2018**, *20*, 19–25. [\[CrossRef\]](#)
3. Bennet, D.J.; MacInnes, C.R.; Suzuki, M.; Uchiyama, K. Autonomous Three-Dimensional Formation Flight for a Swarm of Unmanned Aerial Vehicles. *J. Guid. Control Dyn.* **2011**, *34*, 1899–1908. [\[CrossRef\]](#)
4. Guo, H.; Pang, J.; Han, L.; Shan, Z. Flight Data Visualization for Simulation & Evaluation: A General Framework. In Proceedings of the 2012 Fifth International Symposium on Computational Intelligence and Design, Hangzhou, China, 28–29 October 2012; pp. 497–502.
5. Huo, C.; Lai, T.; Sun, T. The Preliminary Study on Multi-Swarm Sharing Particle Swarm Optimization: Applied to UAV Path Planning Problem. In Proceedings of the 2011 IEEE Congress of Evolutionary Computation, New Orleans, LA, USA, 5–8 June 2011; pp. 1770–1776.
6. Cho, D.-H.; Moon, S.T.; Jang, J.T.; Rew, D.-Y. Collision Avoidance Maneuver Design for the Multiple Indoor UAV by Using AR. Drone. *J. Korean Soc. Aeronaut. Space Sci.* **2014**, *42*, 752–761. [\[CrossRef\]](#)
7. Mellinger, D.; Shomin, M.; Michael, N.; Kumar, V. Cooperative Grasping and Transport Using Multiple Quadrotors. In *Distributed Autonomous Robotic Systems: The 10th International Symposium*; Martinoli, A., Mondada, F., Correll, N., Mermoud, G., Egerstedt, M., Hsieh, M.A., Parker, L.E., Støy, K., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 545–558, ISBN 978-3-642-32723-0.
8. Park, M.; Kang, S.; Lee, J.; Ko, G.; Oh, K.; Ahn, H.; Choi, Y.; Son, J. Realization of Distributed Formation Flying Using a Group of Autonomous Quadcopters and Application to Visual Performance Show. In Proceedings of the 2016 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific), Busan, Korea, 1–4 June 2016; pp. 877–882.
9. Vásárhelyi, G.; Virágh, C.; Somorjai, G.; Tarcai, N.; Szörenyi, T.; Nepusz, T.; Vicsek, T. Outdoor Flocking and Formation Flight with Autonomous Aerial Robots. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 3866–3873.
10. Ars Electronica. Available online: <https://ars.electronica.art/news/de/> (accessed on 23 January 2021).
11. Balinski, M.L. A Competitive (Dual) Simplex Method for the Assignment Problem. *Math. Program.* **1986**, *34*, 125–141. [\[CrossRef\]](#)

12. Barr, R.S.; Glover, F.; Klingman, D. The Alternating Basis Algorithm for Assignment Problems. *Math. Program.* **1977**, *13*, 1–13. [\[CrossRef\]](#)
13. Hung, M.S.; Rom, W.O. Solving the Assignment Problem by Relaxation. *Oper. Res.* **1980**, *28*, 969–982. [\[CrossRef\]](#)
14. Kuhn, H.W. The Hungarian Method for the Assignment Problem. In *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*; Jünger, M., Liebling, T.M., Naddef, D., Nemhauser, G.L., Pulleyblank, W.R., Reinelt, G., Rinaldi, G., Wolsey, L.A., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 29–47. ISBN 978-3-540-68279-0.
15. McGinnis, L.F. Implementation and Testing of a Primal-Dual Algorithm for the Assignment Problem. *Oper. Res.* **1983**, *31*, 277–291. [\[CrossRef\]](#)
16. Eberhardt, S.P.; Daud, T.; Kerns, D.A.; Brown, T.X.; Thakoor, A.P. Competitive Neural Architecture for Hardware Solution to the Assignment Problem. *Neural Netw.* **1991**, *4*, 431–442. [\[CrossRef\]](#)
17. Avis, D.; Devroye, L. An Analysis of a Decomposition Heuristic for the Assignment Problem. *Oper. Res. Lett.* **1985**, *3*, 279–283. [\[CrossRef\]](#)
18. Burkard, R.; Dell’Amico, M.; Martello, S. *Assignment Problems (Other Titles in Applied Mathematics)*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2012; ISBN 978-1-61197-222-1.
19. Chopra, S.; Notarstefano, G.; Rice, M.; Egerstedt, M. A Distributed Version of the Hungarian Method for Multirobot Assignment. *IEEE Trans. Robot.* **2017**, *33*, 932–947. [\[CrossRef\]](#)
20. Zheng, X.; Chunyao, M.A. An Intelligent Target Detection Method of UAV Swarms Based on Improved KM Algorithm. *Chin. J. Aeronaut.* **2020**. [\[CrossRef\]](#)
21. Mirzaeinia, A.; Hassanalian, M. Hassanalian Minimum-Cost Drone-Nest Matching through the Kuhn-Munkres Algorithm in Smart Cities: Energy Management and Efficiency Enhancement. *Aerospace* **2019**, *6*, 125. [\[CrossRef\]](#)
22. Ismail, S.; Sun, L. Decentralized Hungarian-Based Approach for Fast and Scalable Task Allocation. In Proceedings of the 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 13–16 June 2017; IEEE: Miami, FL, USA, 2017; pp. 23–28.
23. Moon, S.; Choi, Y.; Kim, D.; Seung, M.; Gong, H. Outdoor Swarm Flight System Based on RTK-GPS. *J. KIISE* **2016**, *43*, 1315–1324. [\[CrossRef\]](#)
24. Ng, K.M.; Johari, J.; Abdullah, S.A.C.; Ahmad, A.; Laja, B.N. Performance Evaluation of the RTK-GNSS Navigating under Different Landscape. In Proceedings of the 2018 18th International Conference on Control, Automation and Systems (ICCAS), PyeongChang, Korea, 17–20 October 2018; pp. 1424–1428.
25. Achtelik, M.; Achtelik, M.; Brunet, Y.; Chli, M.; Chatzichristofis, S.; Decotignie, J.; Doth, K.; Fraundorfer, F.; Kneip, L.; Gurdan, D.; et al. SFly: Swarm of Micro Flying Robots. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012; pp. 2649–2650.
26. Meier, L.; Honegger, D.; Pollefeys, M. PX4: A Node-Based Multithreaded Open Source Robotics Framework for Deeply Embedded Platforms. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 25–30 May 2015; pp. 6235–6240.
27. Vlase, S.; Marin, M.; Öchsner, A.; Scutaru, M.L. Motion Equation for a Flexible One-Dimensional Element Used in the Dynamical Analysis of a Multibody System. *Contin. Mech. Thermodyn.* **2019**, *31*, 715–724. [\[CrossRef\]](#)
28. Abd-Elaziz, E.; Marin, M.; Othman, M. On the Effect of Thomson and Initial Stress in a Thermo-Porous Elastic Solid under G-N Electromagnetic Theory. *Symmetry* **2019**, *11*, 413. [\[CrossRef\]](#)
29. Nascimento, T.P.; Saska, M. Position and Attitude Control of Multi-Rotor Aerial Vehicles: A Survey. *Annu. Rev. Control* **2019**, *48*, 129–146. [\[CrossRef\]](#)
30. Open Source Autopilot for Drones. Available online: <https://px4.io/> (accessed on 23 January 2021).
31. Koubaa, A.; Allouch, A.; Alajlan, M.; Javed, Y.; Belghith, A.; Khalgui, M. Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. *IEEE Access* **2019**, *7*, 87658–87680. [\[CrossRef\]](#)
32. LinkedIn Is There a Limit to How Many Devices Can Connect to a Wi-Fi Network? Available online: <https://www.lifewire.com/how-many-devices-can-share-a-wifi-network-818298> (accessed on 23 January 2021).
33. Celtek, S.A.; Durdu, A.; Kurnaz, E. Design and Simulation of the Hierarchical Tree Topology Based Wireless Drone Networks. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–30 September 2018; pp. 1–5.
34. Cui, Q.; Liu, P.; Wang, J.; Yu, J. Brief Analysis of Drone Swarms Communication. In Proceedings of the 2017 IEEE International Conference on Unmanned Systems (ICUS), Beijing, China, 27–29 October 2017; pp. 463–466.
35. Laufer, R.; Kleinrock, L. The Capacity of Wireless CSMA/CA Networks. *IEEEACM Trans. Netw.* **2016**, *24*, 1518–1532. [\[CrossRef\]](#)
36. El-Sherif, A.A.; Liu, K.J.R. Throughput Analysis for Cooperation in Random Access Networks. In Proceedings of the 2010 IEEE Wireless Communication and Networking Conference, Sydney, Australia, 18–21 April 2010; pp. 1–6.
37. SungTae, M.; Doyoon, K.; Dongoo, L. Outdoor Swarm Flight System Based on the RTK-GPS. *J. KIISE* **2020**, *47*, 328–334. [\[CrossRef\]](#)
38. Marcus, D.A. *Graph Theory: A Problem Oriented Approach*, 2nd UK ed.; Mathematical Association of America: Washington, DC, USA, 2011; ISBN 978-0-88385-772-4.
39. Burkard, R.E.; Çela, E. Linear Assignment Problems and Extensions. In *Handbook of Combinatorial Optimization: Supplement Volume A*; Du, D.-Z., Pardalos, P.M., Eds.; Springer US: Boston, MA, USA, 1999; pp. 75–149. ISBN 978-1-4757-3023-4.
40. Chen, F.; Chen, N.; Mao, H.; Hu, H. The Application of Bipartite Matching in Assignment Problem. *arXiv* **2019**, arXiv:1902.00256.

-
41. Munkres, J. Algorithms for the Assignment and Transportation Problems. *J. Soc. Ind. Appl. Math.* **1957**, *5*, 32–38. [[CrossRef](#)]
 42. Hall, P. On Representatives of Subsets. *J. Lond. Math. Soc.* **1935**, *s1–s10*, 26–30. [[CrossRef](#)]
 43. Vahrenhold, J. Line-Segment Intersection Made in-Place. *Comput. Geom.* **2007**, *38*, 213–230. [[CrossRef](#)]
 44. Drone Light Shows Powered by Intel. Available online: <https://www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html> (accessed on 23 January 2021).
 45. Intel®Core™ I7-8550U Processor. Available online: <https://www.intel.com/content/www/us/en/products/processors/core/i7processors/i7-8550u.html> (accessed on 6 February 2021).